Counting CTL

François Laroussinie, Antoine Meyer, Eudes Pétonnet

▶ **To cite this version:**

François Laroussinie, Antoine Meyer, Eudes Pétonnet. Counting CTL. FoSSaCS 2010, Mar 2010, Paphos, Cyprus. pp.206-220, 10.1007/978-3-642-12032-9_15 . hal-00681263

HAL Id: hal-00681263
https://hal.science/hal-00681263

Submitted on 21 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Counting CTL

François Laroussinie[1]*, Antoine Meyer[2], and Eudes Petonnet[1]

[1] LIAFA, Université Paris Diderot – Paris 7 & CNRS UMR 7089, France
{Francois.Laroussinie,Eudes.Petonnet}@liafa.jussieu.fr
[2] LIGM, Université Paris Est – Marne-la-Valle & CNRS UMR 8049, France
Antoine.Meyer@univ-mlv.fr

Abstract. This paper presents a range of quantitative extensions for the temporal logic CTL. We enhance temporal modalities with the ability to constrain the number of states satisfying certain sub-formulas along paths. By selecting the combinations of Boolean and arithmetic operations allowed in constraints, one obtains several distinct logics generalizing CTL. We provide a thorough analysis of their expressiveness and of the complexity of their model-checking problem (ranging from P-complete to undecidable).

1 Introduction

Among the existing approaches to the formal verification of automated systems, model checking [7, 17] aims at automatically establishing the validity of a certain formal specification (modeled as a formula in a suitable logic) over the system under study (modeled for instance as a finite automaton). This set of techniques is now well-established and successful, with many real-world applications.

To formalize the specification of temporal properties, for instance in the case of reactive systems, temporal logics (TL) were proposed thirty years ago [16] and widely studied since. They are today used in many model-checking tools. There exists a wide variety of temporal logics, differing for instance by the models over which formulas are interpreted or by the kind of available temporal modalities. Two well-known examples are LTL in the linear-time framework (where formulas are interpreted over infinite runs) and CTL for the branching-time case (where formulas are interpreted over states of Kripke structures). See [8] for a survey of classical temporal logics for systems specification.

Temporal logics have been extended in various ways in order to increase their expressive power. For example, while LTL and CTL only contain future operators, it is also possible to consider past-time modalities to express properties of the past of a run. One can also extend temporal logics with regular expressions (see for instance [19, 10]). Other extensions were proposed to handle *quantitative* aspects of systems. For example, some logics can contain timing constraints to specify that some event P_1 has to occur less than 10 time units before another event P_2. Such temporal logics, like TCTL [2, 9], have been especially studied in

* Partly supported by ANR project DOTS and project QUASIMODO (FP7-ICT).

the framework of timed model checking. Another quantitative extension consists in *probabilistic* logics where one can specify probability bounds over the truth of some property (see for example [4]).

We propose several extensions of CTL with constraints over the number of states satisfying certain sub-formulas along runs. For example, considering a model for an ATM, we can express the property "whenever the PIN is locked, at least three erroneous attempts have been made" by: $\neg \mathsf{EF}_{[\sharp \texttt{error} \leq 2]} \texttt{lock}$ (one cannot reach a state where the PIN is locked but less than two errors have occurred). Similarly, $\neg \mathsf{EF}_{[\sharp \texttt{error} \geq 3]} \texttt{money}$ states that three mistakes forbid cash retrieval. We use subscripts on the temporal modality (as in TCTL) to associate a constraint with the run for which the modality holds. Note that these two properties could be clearly stated in CTL by nesting E_U_ modalities, but the resulting formulas would probably be too big to be easily handled by the user of a model checker. For each extension we consider, we study its expressiveness compared to CTL. In some cases, there is no formal gain of expressiveness because there exist natural translations to obtain equivalent CTL formulas, but these extensions are often *exponentially more succinct* than CTL: they allow writing concise specifications that would require formulas of exponentially larger size in CTL. In other cases, we show that adding some constraints increases the expressive power of CTL.

We consider the model checking problem for various sets $\mathcal{C}$ of constraints, and denote by $\mathsf{CCTL}_{\mathcal{C}}$ the corresponding extension of CTL. We show that polynomial-time algorithms exist when considering Until modalities with constraints of the form[3] $\sum_i \sharp\varphi_i \sim c$ with $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. Additionally allowing Boolean combinations of such constraints or integer coefficients in the sum (or both) makes model checking Δ_2^{P}-complete. We also consider the case of diagonal constraints $\sharp\varphi - \sharp\psi \sim c$ and their more general form $\sum_i \pm\sharp\varphi_i \sim c$ with $c \in \mathbb{Z}$ and show that model checking can be done in polynomial time. However, allowing Boolean combinations of such constraints leads to undecidability. Finally we define a version of CCTL with freeze variables and show that it induces a complexity blow-up: model checking becomes PSPACE-complete.

Several existing works provide related results. In [10], an extension of LTL with a kind of regular expressions containing quantitative constraints over the number of occurrences of sub-expressions is presented. This extension yields algorithms whose time complexity is exponential in the size of formulas and the *value* of integer constants. In [11], extensions of CTL are defined including parameters in constraints. One of these formalisms, namely GPCTL, allows one to express properties with constraints defined as positive Boolean combinations of sums of the form $\sum_i P_i \leq c$ where every P_i is an atomic proposition. Model-checking E_U_ formulas with such a constraint is shown to be NP-complete and a polynomial algorithm is given for a restricted logic (with parameters). In [20], a branching-time temporal logic with general counting constraints (using a variant of freeze variables) is defined to specify event-driven real-time systems. To

[3] For complexity results, we always assume that integer constants are encoded in binary.

obtain decidability, they restrict the analysis to systems verifying some bounded progress condition. In [6, 5], extensions of LTL and CTL with Presburger constraints over the number of states satisfying some formulas are considered, for some class of infinite state processes. The complexity of these problems is much higher than the cases we are concerned with. Finally there also exist timed extensions of CTL interpreted over Kripke structures (see for instance [9]).

The paper is organized as follows. In Section 2, we introduce the definitions of the main formalisms we will use. In Section 3, we show that several of our proposed logics are not more expressive than the classical CTL, yet exponentially more succinct. In Section 4, we address the model-checking problem and provide exact complexity results for most of the logics we introduce. Finally we present in Section 5 a different logic with freeze variables, together with the complexity of its model-checking problem.

2 Definitions

2.1 Models

Let AP be a set of atomic propositions. In branching-time temporal logics, formulas are interpreted over states of Kripke structures.

Definition 1. *A* Kripke structure *(or KS) $\mathcal{S}$ is a tuple $\langle Q, R, \ell \rangle$ where Q is a finite set of states, $R \subseteq Q \times Q$ is a total accessibility relation*[4] *and $\ell : Q \to 2^{AP}$ is a labelling of states with atomic propositions.*

A run ρ of $\mathcal{S}$ is an infinite sequence of states $q_0 q_1 q_2 \ldots$ such that $(q_i, q_{i+1}) \in R$ for every i. We use $\rho(i)$ to denote the state q_i and $\rho_{|i}$ to denote the prefix $q_0 \cdots q_i$ of ρ. $\mathsf{Runs}(q)$ denotes the set of runs starting from some state $q \in Q$. We write $\sigma \leq \rho$ when σ is a prefix of ρ.

We will also consider *Durational Kripke Structures* (DKS), where an integer duration is associated with every transition. Thus for a DKS $\mathcal{S} = \langle Q, R, \ell \rangle$, we have $R \subseteq Q \times \mathbb{N} \times Q$. The duration of a transition is also called a weight or a cost. We use $\mathrm{DKS}^{0/1}$ to denote the DKSs where the durations belong to $\{0, 1\}$. The notion of duration is naturally extended to finite runs of DKSs.

2.2 Counting CTL

We define several extensions of CTL able to express constraints over the number of times certain sub-formulas are satisfied along a run.

Definition 2. *Given a set of atomic propositions AP and a set of constraints $\mathcal{C}$, we define:*

$$CCTL_{\mathcal{C}} \ni \varphi, \psi \ ::= \ P \mid \varphi \wedge \psi \mid \neg\varphi \mid E\varphi U_{[C]}\psi \mid A\varphi U_{[C]}\psi$$

[4] By *total* relation, we mean a relation $R \subseteq Q \times Q$ such that $\forall p \in Q, \exists q \in Q, (p, q) \in R$.

where $P \in \mathsf{AP}$ and $C \in \mathcal{C}$. The sets of constraints we consider are defined as follows, with $l, k \in \mathbb{N}$, $k' \in \mathbb{Z}$ and $\sim \in \{<, \leq, =, \geq, >\}$. First we have the sets of atomic *constraints:*

$$\begin{array}{rll}
\mathcal{C}_0 \ni C & ::= \sharp\varphi \sim k & \text{with } \varphi \in \mathsf{CCTL}_{\mathcal{C}_0} \\
\mathcal{C}_1 \ni C & ::= (\Sigma_{i=1}^{l} \sharp\varphi_i) \sim k & \text{with } \varphi_i \in \mathsf{CCTL}_{\mathcal{C}_1} \\
\alpha\mathcal{C}_1 \ni C & ::= (\Sigma_{i=1}^{l} \alpha_i \cdot \sharp\varphi_i) \sim k & \text{with } \varphi_i \in \mathsf{CCTL}_{\alpha\mathcal{C}_1}, \alpha_i \in \mathbb{N} \\
\mathcal{C}_2 \ni C & ::= (\sharp\varphi - \sharp\psi) \sim k' & \text{with } \varphi, \psi \in \mathsf{CCTL}_{\mathcal{C}_2} \\
\mathcal{C}_3 \ni C & ::= (\Sigma_{i=1}^{l} \pm \cdot \sharp\varphi_i) \sim k' & \text{with } \varphi_i \in \mathsf{CCTL}_{\mathcal{C}_3} \\
\alpha\mathcal{C}_3 \ni C & ::= (\Sigma_{i=1}^{l} \alpha_i \cdot \sharp\varphi_i) \sim k' & \text{with } \varphi_i \in \mathsf{CCTL}_{\alpha\mathcal{C}_3}, \alpha_i \in \mathbb{Z}
\end{array}$$

Let $\mathcal{L}_a$ be the set $\{\mathcal{C}_0, \mathcal{C}_1, \alpha\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \alpha\mathcal{C}_3\}$. We also consider the set of constraints $\mathcal{B}(\mathcal{C})$ for every $\mathcal{C} \in \mathcal{L}_a$, defined as the set of Boolean combinations of atomic constraints in $\mathcal{C}$ with sub-formulas in $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C})}$. We use $\mathcal{L}_b$ for $\{\mathcal{B}(\mathcal{C}) \,|\, \mathcal{C} \in \mathcal{L}_a\}$. Finally $\mathcal{L}_{cons} \stackrel{def}{=} \mathcal{L}_a \cup \mathcal{L}_b$.

We make use of the standard abbreviations $\vee, \Rightarrow, \Leftrightarrow, \bot, \top$, as well as the additional modalities $\mathsf{EF}_{[C]}\varphi \stackrel{\text{def}}{=} \mathsf{E}\top\mathsf{U}_{[C]}\varphi$, $\mathsf{AF}_{[C]}\varphi \stackrel{\text{def}}{=} \mathsf{A}\top\mathsf{U}_{[C]}\varphi$, and their duals $\mathsf{AG}_{[C]}\varphi \stackrel{\text{def}}{=} \neg\mathsf{EF}_{[C]}\neg\varphi$ and $\mathsf{EG}_{[C]}\varphi \stackrel{\text{def}}{=} \neg\mathsf{AF}_{[C]}\neg\varphi$. Any formula occurring in a constraint C associated with a modality in Φ is considered as a sub-formula of Φ. The size $|\Phi|$ of Φ takes the size of these constraints and their sub-formulas into account, assuming that integer constants are encoded in *binary* (unless explicitly stated otherwise). The DAG-size of Φ is the total number of distinct sub-formulas of Φ. As model-checking algorithms compute only once the truth value of a sub-formula, this is generally more relevant to the complexity of model-checking.

The semantics of $\mathsf{CCTL}_{\mathcal{C}}$ formulas (with $\mathcal{C} \in \mathcal{L}_{cons}$) is defined over Kripke structures as follows:

Definition 3. *The following clauses define the conditions for a state q of some KS $\mathcal{S} = \langle Q, R, \ell \rangle$ to satisfy a $\mathsf{CCTL}_{\mathcal{C}}$ formula φ – written $q \models_{\mathcal{S}} \varphi$ – by induction over the structure of φ (we omit Boolean modalities):*

$$\begin{array}{lll}
q \models_{\mathcal{S}} \mathsf{E}\varphi\mathsf{U}_{[C]}\psi & \textit{iff } \exists \rho \in \mathsf{Runs}(q),\ \exists i \geq 0, & \rho(i) \models_{\mathcal{S}} \psi,\ \rho_{|i-1} \models_{\mathcal{S}} C, \\
 & & \textit{and } \forall 0 \leq j < i,\ \rho(j) \models_{\mathcal{S}} \varphi \\
q \models_{\mathcal{S}} \mathsf{A}\varphi\mathsf{U}_{[C]}\psi & \textit{iff } \forall \rho \in \mathsf{Runs}(q),\ \exists i \geq 0, & \rho(i) \models_{\mathcal{S}} \psi,\ \rho_{|i-1} \models_{\mathcal{S}} C, \\
 & & \textit{and } \forall 0 \leq j < i,\ \rho(j) \models_{\mathcal{S}} \varphi
\end{array}$$

Let $\mathcal{C} \in \mathcal{L}_{cons}$ be a set of constraints and C be a constraint in $\mathcal{C}$, the semantics of $\rho_{|i} \models_{\mathcal{S}} C$ is based on the interpretation of $\sharp\varphi$ over $\rho_{|i}$, denoted by $|\rho_{|i}|_{\varphi}$ and defined as: $|\rho_{|i}|_{\varphi} \stackrel{def}{=} |\{j \mid 0 \leq j \leq i \wedge \rho(j) \models_{\mathcal{S}} \varphi\}|$. Given these values, C is interpreted in a natural way.

In the following we omit the subscript $\mathcal{S}$ for $\models$ when no confusion is possible. We use $\equiv$ to denote the standard equivalence between formulas.

Remark 1. The classical X operator ("neXt") can be expressed in $\mathsf{CCTL}_{\mathcal{C}_0}$ as $\mathsf{EX}\varphi \equiv \mathsf{EF}_{[\sharp\top=1]}\varphi$, and that basic constraints in $\mathcal{C}_0$ can be expressed in $\mathcal{C}_2$ because

$\sharp\varphi \equiv \sharp\varphi - \sharp\bot$. Moreover, for $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C})}$ with $\mathcal{C} \in \mathcal{L}_a$, since $\varphi \mathsf{U}_{[C]} \psi \equiv \mathsf{F}_{[C \wedge \sharp(\neg\varphi)=0]}\psi$ the modality F is sufficient to define U; thus such a logic $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C})}$ can also be built from atomic propositions using Boolean operators and modalities $\mathsf{EF}_{[C]}\varphi$ and $\mathsf{AF}_{[C]}\varphi$ (or $\mathsf{EG}_{[C]}\varphi$). Note that all these translations are succinct (linear in the size of formulas) and do not have any impact on complexity results.

Remark 2. The related temporal logic TCTL [2], whose semantics are defined over *timed* models, allows one to label temporal modalities with duration constraints. For instance, one may write $\varphi \mathsf{U}_{<k} \psi$ to express the fact that φ is consistently true until, before k time units have elapsed, ψ eventually holds. When all transitions in a durational Kripke structure have duration 1 (i.e. the duration of any run is equal to its length), TCTL (or RTCTL in [9]) formulas can be directly coded into the logic $\mathsf{CCTL}_{\mathcal{C}_0}$ by only using the sub-formula $\top$ inside constraints. A similar coding is also possible when one uses a proposition *tick* to mark time elapsing as in [15].

2.3 Examples of CCTL formulas.

Consider a model for an ATM, whose atomic propositions include `money`, `reset` and `error`, with the obvious meaning. To specify that it is not possible to get money when three mistakes are made in the same session (*i.e.* with no intermediate reset), we can use the formula $\mathsf{AG}\big(\neg \mathsf{EF}_{[\sharp\mathtt{error} \geq 3 \wedge \sharp\mathtt{reset}=0]}\mathtt{money}\big)$ that belongs to $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$. Note that this could also be expressed by the $\mathsf{CCTL}_{\mathcal{C}_0}$ formula $\mathsf{AG}\big(\neg \mathsf{E}(\neg\mathtt{reset})\mathsf{U}_{[\sharp\mathtt{error}\geq 3]}\mathtt{money}\big)$.

Consider a mutual exclusion algorithm with n processes trying to reach their critical section. We can specify that it verifies the bounded waiting property with bound 10 (*i.e.* when a process P tries to reach its CS, then at most 10 other processes can reach theirs before P does) by the following $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_1)}$ formula: $\mathsf{AG} \bigwedge_i \big(\mathtt{request}_i \Rightarrow \neg \mathsf{EF}_{[\sum_{j\neq i} \sharp\mathtt{CS}_j > 10 \wedge \sharp\mathtt{CS}_i = 0]} \top\big)$.

$\mathsf{AG}_{[\sharp\mathtt{send} - \sharp\mathtt{receive} < 0]}\bot$ belongs to $\mathsf{CCTL}_{\mathcal{C}_2}$ and states that along any finite run, the number of `receive` events cannot exceed the number of `send` events.

Quantitative constraints can also be useful for fairness properties. For example $\mathsf{AG}\,\mathsf{AF}_{[\bigwedge_i 5 \leq \sharp\varphi_i \leq 10]}\top$ expresses that the φ_i's occur infinitely often along every run (as stated with the CTL formula $\bigwedge_i(\mathsf{AG}\ \mathsf{AF}\ \varphi_i)$) but it also ensures some constraint on the number of states satisfying the φ_i's along every execution: for example, it is not possible to have a sub-run where φ_1 holds for 11 states and φ_2 holds only for 4 states.

Note that with $\mathsf{CCTL}_{\alpha\mathcal{C}_3}$ one can express properties over the ratio of two kinds of states along a run. For example, $\mathsf{EF}_{[100\cdot\sharp\mathtt{error} - \sharp\top < 0]}P$ is true when there is a path leading to P such that the rate of `error` states is less than 1 percent. Thus constraints of the form "$\frac{\sharp P}{\sharp P'} \sim k$" can easily be expressed with this logic.

Finally note that we can use any temporal formula inside a constraint (and not only atomic propositions). For example, $\mathsf{AG}(\mathsf{EF}_{[\sharp(\mathsf{EX}\mathtt{alarm}) \leq 5]}\mathtt{init})$ states that it is always possible to reach `init` with a path along which at most 5 states have a successor satisfying `alarm`.

These examples illustrate the ability of CCTL formulas to state properties over the portion of a run leading to some state. A similar kind of properties could also be expressed with past-time modalities (like S or F^{-1}), but unlike these modalities our constraints cannot easily describe the ordering of events in the past: they "only" allow to count the number of occurrences of formulas. We will see in the next sections that our extensions do not always induce a complexity blow-up, while model-checking $\mathsf{CTL}+\mathsf{F}^{-1}$ is known to be PSPACE-complete [14].

3 CCTL expressiveness and succinctness

When comparing two logics, the first question which comes to mind is the range of properties they can be used to define, in other words their *expressiveness*. When they turn out to be equally expressive, a natural way to distinguish them is then to ask *how concisely* each logic can express a given property. This is referred to as *succinctness*, and is also relevant when studying the complexity of model-checking for instance, since it may considerably influence the size of a formula required to express a given property, and hence the time required to model-check it. In this section we study the expressiveness of the different logics defined in the previous section, and provide partial results and comments about their respective succinctness with respect to CTL.

3.1 Expressiveness

We first show that only allowing boolean combinations and positive sums in constraints does not allow CCTL to express more properties than CTL.

Proposition 4. *Any $CCTL_{\mathcal{B}(\alpha\mathcal{C}_1)}$ formula can be translated into CTL.*

Proof (sketch). Let Φ be a $\mathsf{CCTL}_{\mathcal{B}(\alpha\mathcal{C}_1)}$ formula of the form $\mathsf{EF}_{[C]}\varphi$ whose constraint C contains n counting arguments $\sharp\varphi_1$ to $\sharp\varphi_n$, each preceded by a multiplicative constant α_i, and m atomic constraints. We inductively translate Φ to CTL by building a family of formulas whose intended meaning (up to technical details) is as follows:

- If constraint C holds with $\sharp\varphi_i = 0$ for all i, then φ may be true immediately.
- Otherwise, successively check for every i which of the φ_i hold in the current state, updating constraint C along the way by decreasing by α_i the constant to which φ_i is compared in Φ.
- Once all φ_i have been scanned, proceed to the next state and reevaluate C for the new values of the constants.

Each of these steps corresponds to a sub-formula of the form $\Phi_{C',i,b}$ in the CTL translation of Φ, where C' is the current constraint to be checked, i is the index of the formula φ_i being scanned, and b is a boolean flag used to enforce termination, $\Phi_{C,0,\bot}$ being the translation of Φ itself. By counting the number of distinct $\Phi_{C',i,b}$, one can show that the DAG-size of $\Phi_{C,0,\bot}$ is $O(n.k^m)$, where k is the maximal constant appearing in Φ. A similar argument holds for formulas of the form $\mathsf{EG}_{[C]}\varphi$ with the same resulting DAG-size. □

Note that the upper bound for the above translation is parametric, and can be interpreted for all variants of CCTL below $\mathsf{CCTL}_{\mathcal{B}(\alpha\mathcal{C}_1)}$. An example of this translation on a $\mathsf{CCTL}_{\alpha\mathcal{C}_1}$ formula is given in the next section. In contrast to this result, introducing subtractions in constraints yields a strict increase in expressiveness.

Proposition 5. *The $\mathsf{CCTL}_{\mathcal{C}_2}$ formula $\varphi = \mathsf{AG}_{[\sharp A - \sharp B < 0]}\bot$ cannot be translated into CTL.*

Proof (sketch). Formula φ (already seen in Sec. 2 with different atomic propositions) states that the number of B-labeled states cannot exceed the number of A-labeled states along any path. As shown by [3] and also presented in [18], the set of models of any CTL formula can be recognized by a finite alternating tree automaton. From such an automaton, one can easily build a finite alternating automaton over words, whose accepted language is the set of all prefixes of branches in models of the formula, seen as words over 2^{AP}.

Suppose there exists a CTL formula φ' equivalent to φ, and let $\mathcal{A}$ be the alternating tree automaton accepting its set of models. As stated above, from $\mathcal{A}$ one can derive a finite alternating automaton recognizing the set of all words over $2^{\{A,B\}}$ labeling a finite prefix of a branch in a model of φ, namely words whose prefixes contain at most as many B's as A's. Since this language is clearly not regular, this leads to a contradiction. □

3.2 Succinctness

Our extensions of CTL come with three main sources of possible concision, which appear to be orthogonal : the encoding of constants in binary, the possibility to use boolean combinations in the constraints, and the use of sums. However, only the first two prove out to yield an exponential improvement in succinctness :

Proposition 6. *For every formula $\Phi \in \mathsf{CCTL}_{\mathcal{C}_1}$ with unary encoding of integers, there exists an equivalent CTL formula of DAG-size polynomial in $|\Phi|$.*

This proposition is a direct consequence of Prop. 4 where m, the number of atomic constraints, is set to 1. For instance, to translate $\Phi = \mathsf{EF}_{[\Sigma_{i=1}^{n}\sharp \mathsf{p}_i = K]}\varphi$, we define $\forall 0 \leq k \leq K$ the family of CTL formulas:

$$
\begin{array}{ll}
\forall 1 \leq i \leq n, 0 \leq j < n, & \Phi_{i,j,k} = (\mathsf{p}_i \wedge \Phi_{i+1,j+1,k}) \vee (\neg \mathsf{p}_i \wedge \Phi_{i+1,j,k}) \\
\forall 1 \leq j \leq k, & \Phi_{n+1,j,k} = \mathsf{EXE}(\neg \bigvee_{i=1}^{n} \mathsf{p}_i)\mathsf{U}((\bigvee_{i=1}^{n} \mathsf{p}_i) \wedge \Phi_{1,0,k-j}) \\
& \Phi_{n+1,k,k} = \mathsf{EXE}(\neg \bigvee_{i=1}^{n} \mathsf{p}_i)\mathsf{U}\varphi \\
\forall j > k, & \Phi_{n+1,j,k} = \bot
\end{array}
$$

By construction, we have $\Phi \equiv \Phi_{1,0,K}$. The size of this family is $O(n.k)$, thus the DAG-size of $\Phi_{1,0,K}$ is also polynomial in $|\Phi|$, even if its literal size is exponential. This example relies on the fact that constants are encoded in unary, to measure the impact of the addition operation in constraints. We now look at the succinctness gap due to the binary encoding of constants.

Proposition 7. *$CCTL_{\mathcal{C}_0}$ can be exponentially more succinct than CTL.*

Proof. In [15], it is shown that the logic TCTL, when interpreted over Kripke structures with a special atomic proposition *tick* used to mark the elapsing of time, can be exponentially more succinct than CTL. More precisely, the TCTL formulas $\mathsf{EF}_{<n}A$ and $\mathsf{EF}_{>n}A$, which are of size $O(\log(n))$ since n is encoded in binary, do not admit any equivalent CTL formula of temporal height (and hence also size) less than n. These formulas express the existence of a path where A eventually holds and less (resp. more) than n clock ticks are seen until then. They are clearly equivalent to the $O(\log(n))$-size $\mathsf{CCTL}_{\mathcal{C}_0}$ formulas $\mathsf{EF}_{[\sharp tick<n]}A$ and $\mathsf{EF}_{[\sharp tick>n]}A$ respectively. □

This exhibits a first aspect in which CCTL logics can be exponentially more succinct than CTL. However, as expressed in the next proposition, another orthogonal feature of the logic may yield a similar blow-up.

Proposition 8. *$CCTL_{\mathcal{B}(\mathcal{C}_0)}$ with unary encoding of integers can be exponentially more succinct than CTL.*

Proof. It was shown by [18, 1] that any CTL formula φ equivalent to the CTL^+ formula $\psi = \mathsf{E}(\mathsf{F}P_0 \wedge \ldots \wedge \mathsf{F}P_n)$ must be of length exponential in n. It turns out ψ is equivalent to the $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ formula $\psi' = \mathsf{EF}_{[\bigwedge_i \sharp P_i \geq 1]}\top$, which entails the result. Note that ψ' only contains the constant 1, which means that this gap cannot be imputed to the binary encoding. □

The intuitive reason for this blow-up is that a CTL formula expressing the property that atomic propositions P_1 to P_n are each seen at least once along a path would have to keep track of all possible interleavings of occurrences of P_i's.

To summarize, we showed that two different aspects of the extensions of CTL presented in this paper, while not increasing the overall expressiveness of the logic, may yield exponential improvements in succinctness We still have to study similar succinctness properties of the remaining CCTL fragments with respect to CTL and to each other.

4 Model checking

4.1 Model checking $\mathsf{CCTL}_{\mathcal{C}_0}$ and $\mathsf{CCTL}_{\mathcal{C}_1}$

It turns out that model-checking $\mathsf{CCTL}_{\mathcal{C}_1}$ is polynomially equivalent to model-checking $\mathsf{CCTL}_{\mathcal{C}_0}$ (or CTL), as both problems are P-complete.

Theorem 9. *The model-checking problem for $CCTL_{\mathcal{C}_1}$ is P-complete.*

Proof. P-hardness comes from the P-hardness of CTL model-checking. For membership in P, we provide polynomial-time procedures to deal with the subformulas $\mathsf{E}\psi\mathsf{U}_{[C]}\psi'$ and $\mathsf{A}\psi\mathsf{U}_{[C]}\psi'$ with $C \stackrel{\text{def}}{=} \Sigma_{i=1}^{l} \sharp\varphi_i \sim k$. Consider a Kripke structure $\mathcal{S} = (Q, R, \ell)$, and inductively assume that the truth values of ψ, ψ'

and φ_i over each state of $\mathcal{S}$ are known: these sub-formulas will be seen as atomic propositions in the following.

To each state q occurring along a path, we associate a cost $|q|_C = |\{i \mid q \models \varphi_i\}|$, and note that the *value* of $|q|_C$ is in $O(|C|)$. This cost is additively extended to paths in the usual way. Deciding the truth value of the path formula $\psi \mathsf{U}_{[C]} \psi'$ over any path ρ verifying $\psi \mathsf{U} \psi'$ then amounts to checking whether there exists a finite prefix $\rho' q$ of ρ such that $|\rho'|_C \sim k$, $q \models \psi'$ and $\forall i \leq |\rho'|, \rho'(i) \models \varphi$.

We reduce this problem to the model-checking of a TCTL formula over a $\mathrm{DKS}^{0/1}$ (DKS with 0/1-durations) for which there exists a polynomial-time algorithm [15]. We build from $\mathcal{S}$ a $\mathrm{DKS}^{0/1}$ $\mathcal{S}' = (Q', R', \ell')$ as follows: for each state $q \in Q$ with $|q|_C = n$, Q' contains $n+1$ additional states $q_0, \ldots, q_n$. R' is then defined as $\{q \xrightarrow{0} q_0 \mid q \in Q\} \cup \{q_i \xrightarrow{1} q_{i+1} \mid q \in Q, i < |q|_C\} \cup \{q_n \xrightarrow{0} q' \mid (q, q') \in R, n = |q|_C\}$. Finally, we set $\ell'(q_i) = \varnothing$ for all $q_i \in Q' \setminus Q$ and $\ell'(q) = \ell(q) \cup \{\mathsf{ok}\}$ for all $q \in Q' \cap Q$, where ok is a new atomic predicate.

To each path $\rho = q\sigma q'$ in $\mathcal{S}$, we associate the path $\tilde{\rho} = qq_0 \ldots q_n \tilde{\sigma} q'$ in $\mathcal{S}'$. It can now be shown by induction on run lengths that ρ satisfies $\psi \mathsf{U}_{[C]} \psi'$ if and only if $\tilde{\rho}$ satisfies the TCTL path formula $(\mathsf{ok} \Rightarrow \psi) \mathsf{U}_{[\sim k]} (\mathsf{ok} \wedge \psi')$. □

Since $\mathsf{CCTL}_{\mathcal{C}_0}$ includes CTL and is included in $\mathsf{CCTL}_{\mathcal{C}_1}$, we get:

Corollary 1. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{C}_0}$ is P-complete.*

4.2 Model-checking $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ and $\mathsf{CCTL}_{\alpha\mathcal{C}_1}$

We now establish the Δ_2^{P}-completeness of model-checking for the fragments $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$, $\mathsf{CCTL}_{\alpha\mathcal{C}_1}$ and $\mathsf{CCTL}_{\mathcal{B}(\alpha\mathcal{C}_1)}$. Let us first recall the definition of the complexity class Δ_2^{P}, one of the classes of the polynomial hierarchy.

Definition 10. $\Delta_2^{\mathsf{P}} = \mathsf{P}^{\mathsf{NP}}$ *is the class of problems solvable in polynomial time with access to an oracle for some* NP*-complete problem.*

We now prove Δ_2^{P}-hardness of the model-checking problem for $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$.

Theorem 11. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ is Δ_2^{P}-hard.*

Proof. We proceed by reduction from the Δ_2^{P}-complete problem SNSAT [12].

Given p families of variables $X_1, \ldots X_p$ with $X_i = \{x_i^1, ..., x_i^m\}$ and a set $Z = \{z_1, \ldots, z_p\}$ of p variables, an instance $\mathcal{I}$ of SNSAT is defined as a collection of p propositional formulas $\varphi_1, \ldots, \varphi_p$ under 3-conjunctive normal form (3-CNF), where each φ_i involves variables in $X_i \cup \{z_1, ..., z_{i-1}\}$, and the values of each z_i is defined as $z_i \stackrel{\text{def}}{=} \exists X_i.\ \varphi_i(z_1, ..., z_{i-1}, X_i)$. The instance $\mathcal{I}$ is positive iff the value of z_p is $\top$. We denote by $v_{\mathcal{I}}$ the unique valuation of variables in Z induced by $\mathcal{I}$.

From $\mathcal{I}$, we define the KS described in Figure 1. Every state is labeled by its name, and in addition every state $\bar{z}_i$ is labeled by some new atomic proposition $P_{\bar{z}}$. We use X to denote the set $X_1 \cup \cdots \cup X_p$ and $\mathcal{V}$ for $X \cup Z$. A path ρ from q_p to q_F describes the valuation v_ρ such that $v_\rho(y) = \top$ if ρ visits state y and $\bot$ if it visits $\bar{y}$ for every variable y in $\mathcal{V}$. We use a $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ formula to ensure

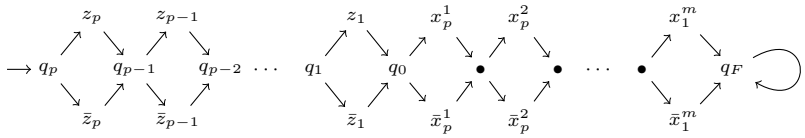

Fig. 1. Kripke structure associated to an SNSAT problem.

that v_ρ coincides with $v_\mathcal{I}$ over Z, that is: $v_\rho(z_i) = \top$ iff $v_\mathcal{I}(z_i) = \top$ for any $i \in \{1, \ldots, p\}$.

Let $\widetilde{\varphi_i}$ be the formula φ_i where every occurrence of the *literal* x is replaced by $\sharp x{=}1$. We define the $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ formula Ψ_0 as $\top$ and for every $1 \leq k \leq p$, Ψ_k as $\mathsf{EX}\big(\mathsf{E}(P_{\bar{z}} \Rightarrow \neg\Psi_{k-1})\mathsf{U}_{[C_k]}q_F\big)$, with $C_k \stackrel{\text{def}}{=} \bigwedge_{l\leq k}\big((\sharp z_l{=}1) \Rightarrow \widetilde{\varphi_l}\big) \wedge \bigwedge_{j=1}^{k}\big((\sharp q{=}j) \Rightarrow \widetilde{\varphi_j}\big)$. The first part of the constraint C_k aims at ensuring that $v_\rho(z_l) = \top$ is witnessed by a valuation for $\{z_1, \ldots, z_{l-1}\} \cup X^l$ satisfying φ_l. The second part ensures the formula φ_j is satisfied by v_ρ when Ψ_k is interpreted from z_j or $\bar{z}_j$ (*i.e.* when the number of qs along the path leading to q_F is j). The formula Ψ_j holds for a state q_i with $i \leq j$ when $v_\mathcal{I}(z_i)$ is $\top$. The embedding of Ψ_{j-1} inside Ψ_j is used to ensure that going through a $\bar{z}_m$ with $i \geq m$ is always necessary w.r.t. $\mathcal{I}$ (*i.e.* there is no way to satisfy the corresponding φ_m):

Lemma 12. *For any $i = 1, \ldots, p$ and $i \leq j \leq p$, we have: $z_i \models \Psi_j \Leftrightarrow v_\mathcal{I}(z_i) = \top$ and $\bar{z}_i \not\models \Psi_j \Leftrightarrow v_\mathcal{I}(z_i) = \bot$*

Now it is sufficient to check whether q_0 satisfies Ψ_p or not, and then deduce the truth value of $v_\mathcal{I}(z_p)$. □

Theorem 13. *The model-checking problem for $\mathsf{CCTL}_{\alpha\mathcal{C}_1}$ is Δ_2^P-hard.*

Proof. We provide a reduction from the model checking problem for TCTL specifications over Durational Kripke structures. TCTL formulas allow to deal with the cost (or duration) of paths (*i.e.* the sum of the weight of every transition occurring along the path). This problem is Δ_2^P-complete [13]. Let $\mathcal{S} = (Q, R_\mathcal{S}, \ell)$ be a DKS. Let W be the set of weights occurring in $\mathcal{S}$. We define the Kripke structure $\mathcal{S}' = (Q', R_{\mathcal{S}'}, \ell')$ as follows:

- $Q' \stackrel{\text{def}}{=} Q \cup \{(q, d, q') \mid \exists (q, d, q') \in R_\mathcal{S}\}$,
- for any $(q, d, q') \in R_\mathcal{S}$, we add $(q, (q, d, q'))$ and $((q, d, q'), q')$ in $R_{\mathcal{S}'}$; and
- $\ell' : Q' \to 2^{\mathsf{AP}'}$ with $\mathsf{AP}' \stackrel{\text{def}}{=} \mathsf{AP} \cup \{\mathsf{ok}\} \cup \{P_d \mid d \in W\}$ – we assume $\mathsf{ok}, P_d \notin \mathsf{AP}$. And we have: $\ell'(q) \stackrel{\text{def}}{=} \ell(q) \cup \{\mathsf{ok}\}$ for any $q \in Q$, and $\ell'(q, d, q') = \{P_d\}$.

Now we can easily see that $q \models_\mathcal{S} \Phi$ with $\Phi \in \mathsf{TCTL}$ is equivalent to $q \models_{\mathcal{S}'} \widetilde{\Phi}$ where $\widetilde{P} \stackrel{\text{def}}{=} P$, $\widetilde{\neg\psi} \stackrel{\text{def}}{=} \neg\widetilde{\psi}$, $\widetilde{\varphi \wedge \psi} \stackrel{\text{def}}{=} \widetilde{\varphi} \wedge \widetilde{\psi}$, $\widetilde{\mathsf{E}\varphi\mathsf{U}_{\sim c}\psi} \stackrel{\text{def}}{=} \mathsf{E}(\mathsf{ok} \Rightarrow \widetilde{\varphi})\mathsf{U}_{[C(\sim c)]}(\mathsf{ok} \wedge \widetilde{\psi})$ and $\widetilde{\mathsf{A}\varphi\mathsf{U}_{\sim c}\psi} \stackrel{\text{def}}{=} \mathsf{A}(\mathsf{ok} \Rightarrow \widetilde{\varphi})\mathsf{U}_{[C(\sim c)]}(\mathsf{ok} \wedge \widetilde{\psi})$ with $C(\sim c) \stackrel{\text{def}}{=} \sum_{d\in W} d \cdot \sharp P_d \sim c$. □

Theorem 14. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{B}(\alpha\mathcal{C}_1)}$ is in Δ_2^P.*

Proof (sketch). Let $\mathcal{S} = \langle Q, R, \ell \rangle$ be a KS. For this proof, we only need to provide NP procedures to deal with sub-formulas of the form $\mathsf{EF}_{[C]}\varphi$ and $\mathsf{EG}_{[C]}\varphi$. First let $\{C_1, \ldots, C_l\}$ be the set of $\alpha\mathcal{C}_1$ constraints occurring in C. Each C_i is of the form $\sum_{j \leq l_i} \alpha^i_j \cdot \sharp\varphi^i_j \sim_i d_i$. And let $d_{\max}$ be the maximal integer constant occurring in C. Now we can present the algorithms:

- $\Phi \stackrel{\text{def}}{=} \mathsf{EF}_{[C]}\psi$: If $q \models \Phi$, then there exists a run ρ starting from q and leading to some q' such that (1) $q' \models \psi$ and (2) ρ without q' satisfies the constraint C. First note that we can assume that the length of ρ is bounded with respect to the model and formula: a sequence of $|Q|$ states contributes for at least 1 to some linear expressions in C and then the length of ρ is in $O(|Q|.2^{|C|})$ due to the binary encoding of the constants. An easy NP algorithm consists in guessing the Parikh image of the transitions in ρ, which can be represented in polynomial size. Moreover it is possible to check (in polynomial time) that q' satisfies ψ, ρ without q' satisfies C, and F_ρ corresponds to a path in $\mathcal{S}$.
- $\Phi \stackrel{\text{def}}{=} \mathsf{EG}_{[C]}\psi$: For this case we have to find an infinite path ρ satisfying the property "if the current prefix satisfies the constraint C, then the next state has to satisfy ψ". Every constraint $C_i \in \alpha\mathcal{C}_1$ in C may change its truth value at most twice along ρ. Therefore ρ can be decomposed in a bounded number of parts over which the truth value of every constraint is constant. As previously, the length of every part is bounded and its Parikh image can be encoded in polynomial size. Moreover it is possible to ensure that the juxtaposition of all ρ_m is correct. □

A direct corollary of Theorems 11, 13 and 14 is:

Corollary 2. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{C}}$ is Δ_2^{P}-complete for each $\mathcal{C} \in \{\alpha\mathcal{C}_1, \mathcal{B}(\mathcal{C}_0), \mathcal{B}(\mathcal{C}_1), \mathcal{B}(\alpha\mathcal{C}_1)\}$.*

4.3 Diagonal constraints

We now show that even if diagonal constraints lead to strictly more expressive logics than CTL, model checking $\mathsf{CCTL}_{\mathcal{C}_2}$ and $\mathsf{CCTL}_{\mathcal{C}_3}$ is not more difficult than model checking CTL itself.

Theorem 15. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{C}_2}$ is P-complete.*

Proof (sketch). P-hardness comes from that of $\mathsf{CCTL}_{\mathcal{C}_0}$ model-checking. Using the fact that $\mathsf{A}\varphi'\mathsf{U}_{[C]}\psi' \equiv \mathsf{AF}_{[C \wedge \sharp\neg\varphi' = 0]}\psi' \equiv \neg\mathsf{EG}_{[C \wedge \sharp\neg\varphi' = 0]}\neg\psi'$, to show membership in P, we only need to provide polynomial-time procedures to verify sub-formulas of the form $\mathsf{E}\varphi'\mathsf{U}_{[C]}\psi'$ and $\mathsf{EG}_{[C \wedge \sharp\varphi' = 0]}\psi'$ with $C \stackrel{\text{def}}{=} \sharp\varphi - \sharp\psi \sim k$. Consider a Kripke structure $\mathcal{S} \stackrel{\text{def}}{=} (Q, R, \ell)$. As previously, we associate a "cost" to each state $q \in Q$. In this case however, $|q|_C$ can only be -1, 0 or 1 depending on the truth values of φ and ψ. Inductively assume that the truth values of φ, ψ, φ' and ψ' over each state of $\mathcal{S}$ are known: these sub-formulas will be seen as atomic propositions in the following. We distinguish the two main cases below:

– $\Phi \stackrel{\text{def}}{=} \mathsf{E}\varphi'\mathsf{U}_{[C]}\psi'$: We consider the weighted and directed graph $G_{\mathcal{S}} = (V, E)$ representing the transition relation of $\mathcal{S}$ restricted to states verifying the formula $\mathsf{E}\varphi'\mathsf{U}\psi'$, where edges are weighted by the cost of their source state and where only edges whose source verifies φ' are considered.
If $C \stackrel{\text{def}}{=} \sharp\varphi - \sharp\psi \leq k$, then the formula holds true on state q if and only if there exists a state q' such that $q' \models_{\mathcal{S}} \psi'$ and either an elementary (i.e. acyclic) path ρ in $G_{\mathcal{S}}$ of weight less than k from q to q', or a path from q to some state q'' appearing on a negative-weight cycle, and from q'' to q'. Using a classical reachability algorithm over $G_{\mathcal{S}}$, one can determine the existence of such paths in polynomial time.
If $C \stackrel{\text{def}}{=} \sharp\varphi - \sharp\psi = k$ with $k \geq 0$, we will compute the relation R_k over V^2 denoting the existence of a run of weight k between states q and q' and simply test whether $(q, q') \in R_k$ for some q' verifying ψ'. Using dichotomy and simple fixed-point computations, we are able to compute R_k in time polynomial in $|\Phi|$, *i.e.* logarithmic in k The treatment of negative weights is omitted.
– $\Phi \stackrel{\text{def}}{=} \mathsf{EG}_{[C \wedge \sharp\varphi' = 0]}\psi'$: We use the weighted and directed graph $G_{\mathcal{S}}$ representing the transition relation of $\mathcal{S}$ where edges are weighted by the cost of their source state, to build a new Kripke structure $\mathcal{S}'$ and a classical CTL formula Ψ such that $\mathcal{S}$ satisfies Φ if and only if $\mathcal{S}'$ satisfies Ψ. □

By combining the techniques used in the previous construction with those used in the proof to Theorem 9, we obtain a similar result for the logic $\mathsf{CCTL}_{\mathcal{C}_3}$.

Corollary 3. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{C}_3}$ is P-complete.*

Proof (sketch). In this setting, each state contributes to the cost of a path by a certain positive or negative number whose absolute value is bounded by some integer d. Similarly to the technique used in the proof of Theorem 9, the idea is to build a durational Kripke structure, this time with weights in $\{-1, 0, 1\}$, by adding intermediate states. Once this DKS is built, relations R_i, R_i^+ and R_i^- may be computed as previously, and the satisfaction of the formula under consideration tested. □

Theorem 16. *The model-checking problem for $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_2)}$ is undecidable.*

Proof (sketch). This is done by reduction from the halting problem of a two-counter machine $\mathcal{M}$ with counters C and D. We define a Kripke structure $\mathcal{S}_{\mathcal{M}}$ with one state to simulate each of $\mathcal{M}$'s instructions, plus some auxiliary states. We use labels φ_X^+ and φ_X^- with $X \in \{C, D\}$ to witness increments and decrements, and additional labels ok_X, ko_X to simulate the positive test "$X = 0$": whenever the counter's value is assumed to be zero, and before simulating the next instruction, the run goes through an auxiliary state labeled ko_X whose unique successor is labeled ok_X. Hence along any run in $\mathcal{S}_{\mathcal{M}}$, a prefix satisfies $\sharp\mathsf{ko}_X > \sharp\mathsf{ok}_X$ right after counter X was deemed equal to zero, and only then. By counting occurrences of these predicates, one can write a $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_2)}$ formula expressing the fact that M is correctly simulated by $\mathcal{S}_{\mathcal{M}}$ and never halts. □

5 Freeze variables

Instead of using counting constraints associated with temporal modalities, we now consider *freeze variables* and explicit constraints inside formulas.

Definition 17. *Given a set of atomic propositions AP and a set of variables V, we define:* $\mathsf{CCTL}^{\mathsf{fv}} \ni \varphi, \psi ::= P \mid \varphi \wedge \psi \mid \neg\varphi \mid z[\psi].\varphi \mid C \mid \mathsf{E}\varphi\mathsf{U}\psi \mid \mathsf{A}\varphi\mathsf{U}\psi$ *where $P \in \mathsf{AP}$ and C is a constraint $\sum_{i=1}^{l} \alpha_i \cdot z_i \sim c$ with $z_i \in V$, $\alpha_i \in \mathbb{N}$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$.*

Intuitively $z[\psi].\varphi$ means that (1) the variable z is reset to zero and associated with the sub-formula ψ (*i.e.* z will evolve like $\sharp\psi$ in the future) and (2) given this semantics for z, φ holds for the current state. We say that an occurrence of some variable z is *free* in φ when this occurrence does not appear in the scope of a reset operator "."; a formula without any free variable is *closed*. For example, the $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ formula $\mathsf{EF}_{[\sharp P \leq 5 \wedge \sharp P' > 2]} P''$ can be expressed in $\mathsf{CCTL}^{\mathsf{v}}$ as the formula $z[P].z'[P'].\mathsf{EF}(z \leq 5 \wedge z' > 2 \wedge P'')$.

A $\mathsf{CCTL}^{\mathsf{v}}$ formula φ is interpreted in a state of a KS extended with a valuation for any free variable in φ and an environment associating a sub-formula to any free variable. We use dom to denote the domain of such functions and $\perp$ to represent undefined values. Given a function f and $x \in \mathsf{dom}(f)$, we use $f[x \leftarrow a]$ to denote the function mapping x to a, and every element y to $f(y)$ if $y \neq x$. Finally let $\mathsf{SubF}(\varphi)$ be the set of all φ sub-formulas.

Given a valuation $v : V \to \mathbb{N} \cup \{\perp\}$ for a set of variables occurring in a $\mathsf{CCTL}^{\mathsf{v}}$ formula φ, and an environment $\varepsilon : V \to \mathsf{SubF}(\varphi) \cup \{\perp\}$ such that $\mathsf{dom}(v) = \mathsf{dom}(\varepsilon)$, and given a finite run π, we define the valuation $(v +_\varepsilon \pi)$ as follows: $(v +_\varepsilon \pi)(z) \stackrel{\text{def}}{=} \perp$ if $z \notin \mathsf{dom}(v)$, and otherwise $(v +_\varepsilon \pi)(z) \stackrel{\text{def}}{=} v(z) + |\{j | 0 \leq j \leq |\pi| \wedge \pi(j) \models \varepsilon(z)\}|$. The semantics of $\mathsf{CCTL}^{\mathsf{v}}$ is defined as follows:

Definition 18. *The following clauses define when a state q of some KS $\mathcal{S} = \langle Q, R, \ell \rangle$ and a valuation v satisfy a $\mathsf{CCTL}^{\mathsf{v}}$ formula φ in an environment ε – written $(q, v) \models_{\mathcal{S},\varepsilon} \varphi$ – by induction over the structure of φ (we omit the cases of Boolean modalities):*

$$
\begin{aligned}
(q, v) &\models_{\mathcal{S},\varepsilon} \mathsf{E}\varphi\mathsf{U}\psi \;\; \textit{iff} \;\; \exists \rho \in \mathsf{Runs}(q), v \models_{\rho,\varepsilon} \varphi\mathsf{U}\psi \\
(q, v) &\models_{\mathcal{S},\varepsilon} \mathsf{A}\varphi\mathsf{U}\psi \;\; \textit{iff} \;\; \forall \rho \in \mathsf{Runs}(q), v \models_{\rho,\varepsilon} \varphi\mathsf{U}\psi \\
(q, v) &\models_{\mathcal{S},\varepsilon} z[\psi].\varphi \;\; \textit{iff} \;\; (q, v[z \leftarrow 0]) \models_{\mathcal{S},\varepsilon[z \leftarrow \psi]} \varphi \\
(q, v) &\models_{\mathcal{S},\varepsilon} \Sigma_{i=1}^{l} \alpha_i \cdot z_i \sim c \;\; \textit{iff} \;\; \Sigma_{i=1}^{l} \alpha_i \cdot v(z_i) \sim c
\end{aligned}
$$

where $v \models_{\rho,\varepsilon} \varphi\mathsf{U}\psi$ iff $\exists i \geq 0$, $(\rho(i), v +_\varepsilon \rho_{|i-1}) \models_{\mathcal{S},\varepsilon} \psi$ and $\forall 0 \leq j < i, (\rho(j), v +_\varepsilon \rho_{|j-1}) \models_{\mathcal{S},\varepsilon} \varphi$.

Theorem 19. *Model checking closed $\mathsf{CCTL}^{\mathsf{v}}$ formulas is* PSPACE*-complete.*

Proof. PSPACE-hardness can be proved by a reduction from QBF. PSPACE-membership is obtained by considering a non-deterministic algorithm working

in polynomial space to decide whether a $\mathsf{CCTL}^{\vee}$ formula holds for a state q within a KS $\mathcal{S}$. The main idea is to encode a configuration (q, v, ε) in polynomial size: this is possible for v since we just have to record the value for the counter z up to $d_{\max} + 1$ where $d_{\max}$ is the maximal constant used in a constraint with z. In order to verify $\mathsf{E}\varphi\mathsf{U}\psi$ – we assume that φ and ψ have already been treated – we guess, from the current configuration (q, v, ε), the next configuration (q', v', ε) and then we verify that there is a transition in $\mathcal{S}$ leading from q to q' such that the valuation v is updated with v' w.r.t. the environment ε. Then it remains to verify that either ψ or φ holds for (q', v') (and in the latter case, guess a new configuration *etc.*). The same holds for EG. The operator $z[\psi].\varphi$ changes the environment ε and resets z to zero. And for any configuration one can decide the truth value of a constraint C. □

6 Conclusion

In several cases (up to $\mathcal{B}(\alpha\mathcal{C}_1)$ constraints) the logics we introduce are not more expressive than CTL, but can concisely express properties which would be difficult to write in that logic. In particular, $\mathsf{CCTL}_{\mathcal{C}_0}$ and $\mathsf{CCTL}_{\mathcal{B}(\mathcal{C}_0)}$ can be exponentially more succinct than CTL. As for the remaining fragments, even though $\mathsf{CCTL}_{\mathcal{C}_2}$ is strictly more expressive than CTL, model-checking remains polynomial up to $\mathsf{CCTL}_{\mathcal{C}_3}$ (complexity results are summarized in Figure 2). Further work on CCTL will include completing the study of succinctness of its fragments with respect to each other and to other logics, looking for an upper complexity bound for the model-checking of $\mathsf{CCTL}_{\alpha\mathcal{C}_3}$, as well as investigating new kinds of constraints and extensions to LTL and CTL^*.

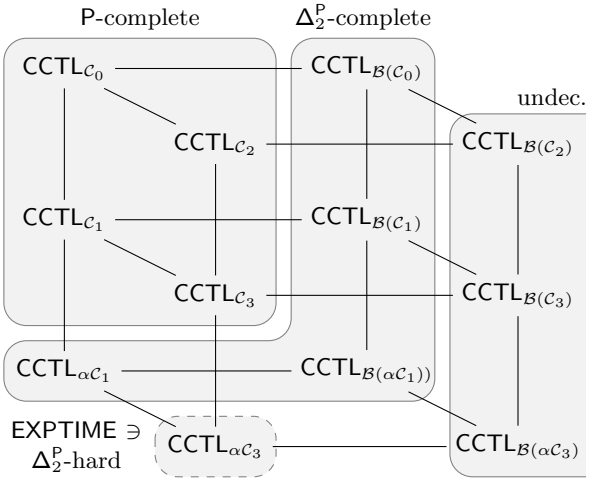

Fig. 2. Summary of model-checking complexity results.

References

1. M. Adler and N. Immerman. An $n!$ lower bound on formula size. *ACM Transactions on Computational Logic*, 4(3):296–314, 2003.
2. R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.
3. O. Bernholtz, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. In *Proc. 6th CAV*, volume 818 of *LNCS*, pages 142–155. Springer, 1994.
4. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *15th FSTTCS*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.
5. A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *Proc. 10th LICS*, pages 123–133. IEEE Comp. Soc. Press, 1995.
6. A. Bouajjani, R. Echahed, and P. Habermehl. Verifying infinite state processes with sequential and parallel composition. In *Proc. 22nd POPL*, pages 95–106, 1995.
7. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.
8. E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science, 1990.
9. E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4(4):331–352, 1992.
10. E. A. Emerson and R. J. Trefler. Generalized quantitative temporal reasoning: An automata-theoretic approach. In *Proc. 7th TAPSOFT*, volume 1214 of *LNCS*, pages 189–200. Springer, 1997.
11. E. A. Emerson and R. J. Trefler. Parametric quantitative temporal reasoning. In *Proc. 14th LICS*, pages 336–343. IEEE Comp. Soc. Press, 1999.
12. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking CTL^+ and $FCTL$ is hard. In *Proc. 4th FoSSaCS*, volume 2030 of *LNCS*, pages 318–331. Springer, 2001.
13. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Efficient timed model checking for discrete-time systems. *Theor. Comput. Sci.*, 353(1-3):249–271, 2006.
14. F. Laroussinie and Ph. Schnoebelen. Specification in CTL+Past for verification in CTL. *Inf. Comput.*, 156(1/2):236–263, 2000.
15. F. Laroussinie, Ph. Schnoebelen, and M. Turuani. On the expressivity and complexity of quantitative branching-time temporal logics. *Theor. Comput. Sci.*, 297(1–3):297–315, 2003.
16. A. Pnueli. The temporal logic of programs. In *Proc. 18th FOCS*, pages 46–57. IEEE Comp. Soc. Press, 1977.
17. J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. 5th Int. Symp. on Programming*, volume 137 of *LNCS*, pages 337–351. Springer, 1982.
18. T. Wilke. CTL^+ is exponentially more succinct than CTL. In *Proc. 19th FSTTCS*, volume 1738 of *LNCS*, pages 110–121. Springer, 1999.
19. P. Wolper. Temporal logic can be more expressive. *Inf. and Control*, 56(1/2):72–99, 1983.
20. J. Yang, A. K. Mok, and F. Wang. Symbolic model checking for event-driven real-time systems. *ACM Transactions on Programming Languages and Systems*, 19(2):386–412, 1997.